

# **CSE 259 - Logic in Computer Science**

## **Fall 2024**

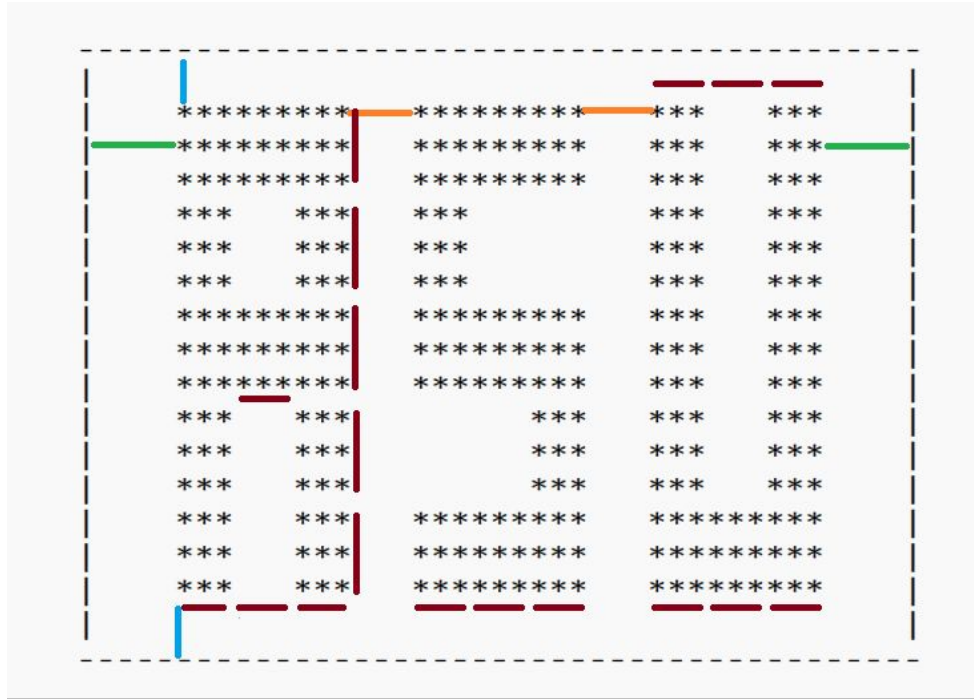
### **Recitation-5**

**Waqar Hassan Khan**



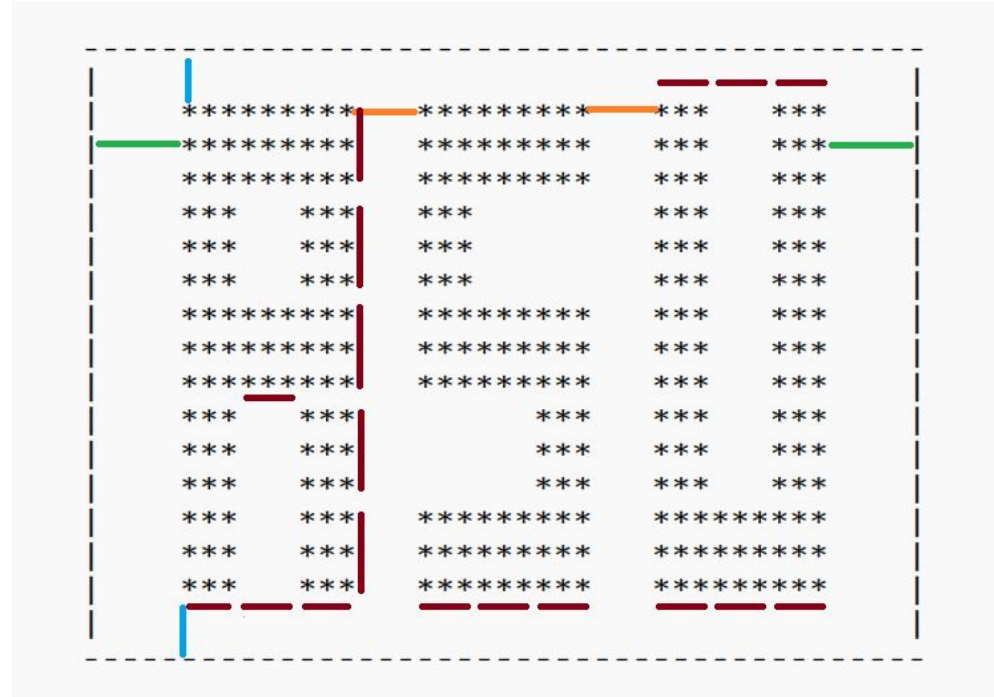
# Project-1

- Use the query ***asu(LeftRightMargin, BottomTopMargin, SpaceBetweenCharacters, FontSize)***
- The output should look like the following



# How to implement

- Print the horizontal top line
- Print vertical lines with spaces for BottomTopMargin number of times
- Print vertical lines with \* and spaces
- Print vertical lines with spaces for BottomTopMargin number of times
- Print the horizontal bottom line



# How to implement

You, 9 minutes ago • fixed bug

```
asu(LeftRightMargin, BottomTopMargin, SpaceBetweenCharacters, FontSize) :-  
  integer(LeftRightMargin), integer(BottomTopMargin), integer(SpaceBetweenCharacters), integer(FontSize),  
  Width is (LeftRightMargin * 2 + SpaceBetweenCharacters * 2 + FontSize * 3 * 3 + 2),  
  Height is (BottomTopMargin * 2 + FontSize * 5),  
  
  drawHorizontalLine('-', Width),  
  nl,  
  drawVerticalLinesWithSpace('|', BottomTopMargin, Width),  
  drawVerticalLinesWithCharacters(LeftRightMargin, BottomTopMargin, SpaceBetweenCharacters, FontSize),  
  drawVerticalLinesWithSpace('|', BottomTopMargin, Width),  
  drawHorizontalLine('-', Width).
```

# Understanding the template

- `drawHorizontalLine('-', Width)` is the same as before. Like we have drawn the rectangle in the last class!
- `drawVerticalLinesWithSpace('|', BottomTopMargin, Width)` draws the vertical lines just with spaces.
- `drawVerticalLinesWithCharacters(LeftRightMargin, BottomTopMargin, SpaceBetweenCharacters, FontSize)` draws the vertical lines with \* and space characters.

# Understanding the template

## **drawVerticalLinesWithCharacters**

- Calculates the width and height of the letters: A, S, and U
- Width would be horizontal segment size x font size:  $3 * \text{FontSize}$
- Height would be vertical segment size x font size:  $5 * \text{FontSize}$
- Initiate a variable called CurrentLine with 0
- This function is like a driver code. Think of an DFS algorithm implementation in C++. We use a driver function to initialize things and then call a recursive function. Same thing will be done here. We are going to call a recursive rule called **draw** from **drawVerticalLinesWithCharacters**

# Understanding the template

```
drawVerticalLinesWithCharacters(LeftRightMargin, BottomTopMargin, SpaceBetweenCharacters, FontSize) :-  
    CurrentLine is 0,  
    TextWidth is FontSize * 3,           You, 22 minutes ago • fixed bug  
    TextHeight is FontSize * 5,  
    draw(LeftRightMargin, SpaceBetweenCharacters, FontSize, CurrentLine, TextWidth, TextHeight).
```

# Understanding the template

**draw(LeftRightMargin, SpaceBetweenCharacters, FontSize, CurrentLine, TextWidth, TextHeight)**

- If CurrentLine is less than the TextHeight then do things to print. Else don't do anything
- Initiate ColumnNumber = 0
- Print | and then space
- Print the necessary characters for the letter A for the current line
- Print space between letters
- Print the necessary characters for the letter S for the current line
- Print space between letters
- Print the necessary characters for the letter U for the current line
- Print space and then |



# Understanding the template

```
draw(LeftRightMargin, SpaceBetweenCharacters, FontSize, CurrentLine, TextWidth, TextHeight) :-  
    CurrentLine >= TextHeight.  
draw(LeftRightMargin, SpaceBetweenCharacters, FontSize, CurrentLine, TextWidth, TextHeight) :-  
    CurrentLine < TextHeight,  
    ColumnNumber is 0,  
    write('|'), drawSymbol(' ', LeftRightMargin),  
    drawA(TextWidth, TextHeight, FontSize, CurrentLine, ColumnNumber),  
    /*-----*/  
    /** WRITE YOUR CODES HERE **/  
    % add spaces here between A and S  
    % drawS  
    % add spaces here between S and U  
    % drawU  
    /*-----*/  
    drawSymbol(' ', LeftRightMargin),  
    write('|'),  
    nl,  
    NextLine is CurrentLine + 1,  
    draw(LeftRightMargin, SpaceBetweenCharacters, FontSize, NextLine, TextWidth, TextHeight).
```

# Understanding the template - draw A

- Use four different rules

# Understanding the template - draw A

## Rule-1

- We draw the letters line by line, i.e., 1st line of A, S, and U then 2nd line, then 3rd line, and so on.
- Before starting to draw each letter, RESET ColumnNumber to 0
- The first rule would just check whether ColumnNumber is greater or equal to the required width of the letter

```
drawA(TextWidth, TextHeight, FontSize, CurrentLine, ColumnNumber) :-  
    ColumnNumber >= TextWidth.
```

# Understanding the template - draw A

## Rule-2

- Covers the left most segment and the rightmost segment
- If FontSize = 3, TextWidth = 9
- Left-most segment has ColumnNumber 0, 1, 2 and Right-most segment has ColumnNumber 6, 7, 8
- So, we generalize the formula in the code

```
drawA(TextWidth, TextHeight, FontSize, CurrentLine, ColumnNumber) :-  
  (  
    (ColumnNumber >= 0, ColumnNumber < FontSize);  
    (ColumnNumber >= FontSize * 2, ColumnNumber < TextWidth )  
  ),  
  drawSymbol('*', FontSize),  
  NextColumn is ColumnNumber + FontSize,  
  drawA(TextWidth, TextHeight, FontSize, CurrentLine, NextColumn).
```



# Understanding the template - draw A

## Rule-3

- Covers the marked segments of the middle segment
- If FontSize = 3, TextWidth = 9 then covers ColumnNumber 3, 4, 5
- CurrentLine can be either  $0 \leq \text{CurrentLine} < 3$  or  $6 \leq \text{CurrentLine} < 9$
- So, we generalize the formula in the code

```
drawA(TextWidth, TextHeight, FontSize, CurrentLine, ColumnNumber) :-  
  (ColumnNumber >= FontSize, ColumnNumber < FontSize * 2),  
  (  
    (CurrentLine >= 0, CurrentLine < FontSize);  
    (CurrentLine >= FontSize * 2, CurrentLine < FontSize * 3)  
  ),  
  drawSymbol('*', FontSize),  
  NextColumn is ColumnNumber + FontSize,  
  drawA(TextWidth, TextHeight, FontSize, CurrentLine, NextColumn).
```



# Understanding the template - draw A

## Rule-4

- Covers the marked segments of the middle segment
- If  $\text{FontSize} = 3$ ,  $\text{TextWidth} = 9$  then covers  $\text{ColumnNumber} 3, 4, 5$
- $\text{CurrentLine}$  can be either  $3 \leq \text{CurrentLine} < 6$  or  $9 \leq \text{CurrentLine} < \text{TextHeight}$
- So, we generalize the formula in the code

```
drawA(TextWidth, TextHeight, FontSize, CurrentLine, ColumnNumber) :-  
  (ColumnNumber >= FontSize, ColumnNumber < FontSize * 2),  
  (  
    (CurrentLine >= FontSize, CurrentLine < 2 * FontSize);  
    (CurrentLine >= FontSize * 3, CurrentLine < TextHeight)  
  ),  
  drawSymbol(' ', FontSize),  
  NextColumn is ColumnNumber + FontSize,  
  drawA(TextWidth, TextHeight, FontSize, CurrentLine, NextColumn).
```



# Understanding the template

```
· /*-----*/  
· /** WRITE YOUR CODES HERE **/  
· % add spaces here between A and S  
· ColumnNumber is 0,  
· % drawS  
· % add spaces here between S and U  
· ColumnNumber is 0,  
· % drawU  
· /*-----*/
```